

Solaris 10 Password History

by: Glenn M. Brunette, Jr., 10/20/2004

<http://www.securitydocs.com/library/2646>

Today's topic is short and sweet: password history. For those who are not aware, the concept of password history is to prevent users from repeatedly selecting the same (set) of passwords over and over again (within some fixed time window). The actual time window will depend on the configuration of the system.

Often, password history is used in combination with password composition rules and password aging. For example, an organization may prohibit user's from selecting a password that is defined in some dictionary list or one that does not meet some minimum set of composition requirements. Further, the same organization may also mandate that user's must change their password every ninety days (but no sooner than one week after the last successful change). The goal here is to require users to select strong passwords, change them regularly and limit their future reuse. Without password history, what would stop a user from repeatedly selecting the same password (or small set of passwords - or even just rotating between two valid passwords) at each password expiration event?

Password history helps to solve this issue by enforcing a policy that says that a user may not reuse any of the last n passwords (where n is defined by the organization). So, for example, by setting both password history and aging settings appropriately, an organization could prevent the reuse of passwords for a significant period of time.

So, let's talk about specifics. New to Solaris 10 is the *HISTORY* parameter in the [/etc/default/passwd](#) file. This new parameter specifies the number of passwords that will be remembered - and consequently compared against a user's new password to see if it had been used before. By default, password history is disabled. If you want to enable password history then you must uncomment the *HISTORY* parameter and assign it a value. This value will indicate the number of passwords that the system should remember. This parameter can take values ranging from 0 to 26. For example:

```
# grep "^HISTORY=" /etc/default/passwd
HISTORY=10
```

In this example, password history has been enabled and a user's last ten passwords will be remembered. So, if I user attempts to re-use a password that is in their history, the change will be denied and the user will be presented with the following message:

```
$ passwd gmb
Enter existing login password:
New Password:
passwd: Password in history list.
```

```
Please try again
New Password:
```

Once password aging has been enabled, it can be disabled by setting the *HISTORY* parameter to 0 or by commenting it out in the [/etc/default/passwd](#) file. Once this is done, all users' prior password history will be discarded at the next password change by any user. (Note that this is a very important point. When

password history is disabled, the entire password history database will be discarded upon the next password change event by any user on the system.)

In my previous posts I have tried to highlight global changes versus settings that can be applied to individual users. In the case of password history, it can only be defined on a global basis. There are no per-user settings for this feature.

Now that I have covered the basics of enabling, configuring and disabling password history, one last question may still be nagging at you - where is the password history database kept? The password history database is stored in the `/etc/security/passhistory` file. This file is an implementation artifact of the password history feature and is not meant to be modified by end users. This file has the following ownership and permissions:

```
-r----- 1 root root 47 Sep 28 22:21 /etc/security/passhistory
```

Those curious enough to poke around this file will find that it contains a list of users (one per line) as well as their last *n* password hashes separated by a colon. For example, the above file has the following contents:

```
# cat /etc/security/passhistory
gmb:TRltRapbB7Eek:l5rXbTq1EJ7bQ:yEB668aaGv5z6:LgbM3LpCsERlA:
```

What is really cool about how password history is implemented in Solaris 10 is that it will work as you change your default password encryption algorithm using the [flexible crypt](#) mechanism (introduced in Solaris 9). For example, after changing my password encryption algorithm from the Solaris default to Blowfish, this is what happens:

```
# grep "^HISTORY=" /etc/default/passwd
HISTORY=10

# grep "^CRYPT_DEFAULT=" /etc/security/policy.conf
CRYPT_DEFAULT=__unix__

# cat /etc/security/passhistory
gmb:aMPK0ug.Syoag:Lp145TNOHmdlM:

# grep "^CRYPT_DEFAULT=" /etc/security/policy.conf
CRYPT_DEFAULT=2a

# grep "^2a" /etc/security/crypt.conf
2a      crypt_bsdbf.so.1

# passwd gmb
New Password:
Re-enter new Password:
passwd: password successfully changed for gmb

# cat /etc/security/passhistory
gmb:$2a$04$A.vGapPSctbmXj3B9hYK..7fkgJqpg3YKXFoOt1T.YLBk0xw5p9E.:aMPK0ug.Syoag:Lp14
5TNOHmdlM:
```

First, I verified that I was using the default password encryption algorithm ("__unix__"). I used the `passwd(1)` command twice (not shown) to define passwords for the *gmb* account. Those password hashes were recorded in the `/etc/security/passhistory` file. I then changed the default password encryption algorithm to Blowfish ("2a") and changed the password for the *gmb* account once more. Each time I changed the password, I used a new, unique password.

Now we are ready for the real test. I will try to select each of the three passwords that are in my password history. Two of the passwords are currently stored in default ("__unix__") format and the other is stored in Blowfish format.

```
$ passwd gmb  
Enter existing login password:  
New Password:  
passwd: Password in history list.
```

```
Please try again  
New Password:  
passwd: Password in history list.
```

```
Please try again New Password:  
passwd: Password in history list.  
Permission denied
```

As you can see, in each case, the password was recognized as having been in my password history. So, it doesn't matter if you simply use the default password algorithm or select another. This functionality will still just work. Don't you just love it!

Well, this one ran a little longer than I had originally thought, but I hope that you found it interesting nonetheless. Check back soon for another installment of lesser known and/or publicized security enhancements to the Solaris 10 OS. I still have a bunch lined up for you!